操作可能なPDGによるプログラミング支援について

藤堂　秀仁† 松田　憲幸‡ 柏原　昭博‡ 平嶋　宗‡ 江澤　義典† 豊田　順一‡
†関西大学工学部 ‡大阪大学産業科学研究所

2K－2

1.はじめに

プログラムの作成過程において、プログラマは変数がどこで定義され、参照されるのかといった変数間の依存関係や文と文の制御の依存関係を意識している。これらの依存関係は、ソースプログラム中では暗黙的であり、かつプログラムを修正するたびに変化する。そのため、プログラマがこれらの依存関係を把握することは必ずしも容易ではない。プログラマのこのような負荷を軽減する方法の1つとして、プログラムの文間や変数間の依存関係を視覚化することが考えられる。その変数間の依存関係や文間の依存関係を視覚化する手法の1つにPDG(Program Dependence Graph)[1][2]がある。

本研究では、このPDGを用いたプログラミング支援環境の設計開発を行っている。本支援環境では、ソースプログラムの変更を動的にPDGに反映させることによって、それぞれの依存関係の変化が把握できるようにする。逆にPDGでの操作をソースプログラムに反映させることによって、依存関係を直接操作するプログラミングを可能にする。

本稿では、特にプログラマによるPDGの操作を可能とする機能について述べる。プログラマがPDGを操作するとき、その対象物であるノードやリンクを変更しただけでは、依存関係でない部分に依存関係のリンクがついたり、依存関係である部分に依存関係のリンクがついていないといったことが起こり、依存関係の整合性に矛盾が生じる。またPDGの変更にともなって、副次的に他のものが変更されることがある。したがって、プログラマによるPDGの操作に対してPDGのどの部分が変化するかを整理する必要がある。変化する部分には、プログラマにとって保存してほしい依存関係などがあることがある。ここではプログラマにとって保存してほしいものを指定させることをPDGの変更に制約を与えるという。このことを可能とするため、各操作に対して制約の候補には何があるかを整理し、与えられた制約に対して操作後のPDGにどのような候補があるかの分類を行う。

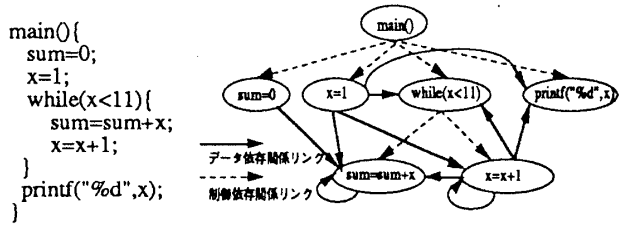
このような分類に基づいてプログラマがPDGを操作することができる環境を設計開発している。

2.PDGに基づくプログラミング支援環境

2.1 PDG(Program Dependence Graph)

PDGは、有向グラフの一種であり、ノードには、ふつうプログラムの各1命令文がはいる。ただし、if文やwhile文といったプログラム制御に関係のある命令文はその命令文の条件部分を1つのノードとする。

リンクには3種類のリンクがある。1つは変数vが定義された文のノードからその変数vが参照されている文のノードへのリンクをデータ依存関係リンクという。2つめは文s1によって文s2が実行されるかされないかが決まる関係があるとき、文s1から文s2へのリンクを制御依存関係リンクという。さらにC言語を対象とした場合、ポインタ変数にアドレスを定義しているノードからそのポインタ変数を参照しているノードへのリンクを考え、これをアドレス依存関係リンクという。



(a) ソースプログラム　(b) PDGの例

図1 PDGの例

例えば、図1(a)のプログラムのPDGは図1(b)のようになる。

2.2 支援機能

本研究において開発しようとしているPDGを用いたプログラム作成過程の支援環境の諸機能について述べる。

1. PDGの動的表示:ソースを変更すると動的にPDGが変わる。これによって逐次、変数の依存関係の変化がわかり、プログラマが変数の依存関係の変化を考慮する負荷が軽減される。
2. PDG上での変更操作:PDGを操作することによって、変更操作後のPDGの候補を出す。これによってプログラマは広い視野でプログラムの作成ができる。
3. PDG変更のソースへのマッピング:変更されたPDGをソースに反映させる。つまり、PDGの操作だけでプログラムの作成が可能となる。
4. プログラムの部分的シミュレーション:プログラムの部分実行を行う。これによってプログラムの一部分がうまく動作するのかをためすことができる。

本稿では、2.のPDG上での変更操作ができる環境の作成について述べる。

3.操作可能なPDG

3.1操作タイプの分類

PDGを操作できる環境を作成する場合、PDGに対するどのような操作をプログラマに許すかを決める必要がある。さらにその操作によって変更される対象物も整理しておかなければならない。表1は、横軸の対象物を縦軸の操作タイプで操作した場合にPDGが変更前と後で何が変わるのかを示したものである。プログラマのできるPDG上の操作としては、ノードやリンクを異なる位置へ動かす移動タイプ、PDG上に新たにノードやリンクを追加する追加タイプ、現在のPDGのノードやリンクの削除する削除タイプ、ノードやリンクの中身を書き換える書き換えタイプがある。

例えば、図1(b)の"x=x+1"のノードを削除する場合ノードだけが削除されるのではなく、このノードに入力されるデータ依存関係リンク、出力されるデータ依存関係リンク、"while(x<11)"から"x=x+1"までの制御依存関係リンクも削除される。

Programming Support with Manipulatable PDG
H. Todo†,N. Matsuda‡,A. Kashihara‡,T. Hirashima‡,
Y. Ezawa† and J. Toyoda‡
†Kansai University
‡The Institute of Scientific and Industrial Research,
Osaka University

表1　操作タイプの分類

操作タイプ＼対象物	ノード	リンク		
		制御依存関係	データ依存関係	アドレス依存関係
移動	CD DD AD	DD AD	DD ノード	DD AD ノード
追加 削除	CD DD AD	CD DD AD	DD ノード	DD AD ノード
書き換え	DD AD		ノード	ノード

注）DD：データ依存関係リンク
CD：制御依存関係リンク
AD：アドレス依存関係リンク

表2 変更タイプに対する制約の分類

対象物＼タイプ		ノード			データ依存関係リンク
		文中に変数の定義と参照がある文	文中で変数が参照されている文	while,if文などのプログラムの制御に関係のある文	
追加	複写	・複写元ノードに入力されるDDの保存 ・複写元ノードから出力されるDDの保存	・複写元ノードに入力されるDDの保存	・複写元ノードに入力されるDDの保存	
追加	新規追加	・追加されるノードの参照変数のDDの保存 ・追加されるノードの定義変数のDDの保存 ・追加されるノードの変数以外の変数のDDの保存	なし	なし	・CDの保存 ・定義側ノードから出力されるDDの保存 ・参照側ノードに入力されるDDの保存
削除		・削除されるノードに入力されるDDの保存 ・削除されるノードに出力されるDDの保存 ・削除されるノード以外のCDの保存	・削除されるノードに入力されるDDの保存 ・削除されるノード以外CDの保存	・削除されるノードに入力されるDDの保存	・CDの保存 ・定義側ノードから出力されるDDの保存 ・参照側ノードに入力されるDDの保存

注）表中のDDはデータ依存関係リンクを表し、CDは制御依存関係リンクを表す。

3.2PDG変更の制約の分類

表1では、プログラマのPDGの操作要求に対してその操作後のPDGが副次的に何が変わってしまうかを示した。このようにPDGの操作によってプログラマの変更されたくない対象物が変化することがある。PDGを操作するとき、変更されたくないものを指定することを制約を与えるという。制約は、大別して制御依存関係の保存とデータ依存関係の保存に分類できる。そこで、プログラマの行おうとしている操作に対して、システムがその操作に応じて制約の候補を提示し、その中から保存させたいものをプログラマに指定させることによって制約を与えさせる。表1は、プログラマのPDG操作に対する表であるが、システム側からすれば、PDGの移動や書き換えは対象物を削除して追加すればよいので、システムの変更のタイプは追加と削除のみを考えればよいことになる。次にシステム側から見た制約の候補には何があるのかを考える。表1は各操作に対して何が変化するかを示したものであるが、変化するもののうち対象物の操作に直接関係のある依存関係について整理したものを制約の候補として考えることにする。表2は対象物を変更のタイプによってどのような制約の候補があるのかを示したものである。例えば、代入文のノードを消去しようとする場合、システムは削除されるノードから出力されるデータ依存リンクを保存するのか、入力されるデータ依存関係リンクを保存するのか、あるいは削除されるノード以外の制御依存関係リンクを保存するのかといった制約の候補を提示する。

3.3PDG変更の候補の分類

PDGを操作する場合、その操作後のPDGが一意に決まるとは限らない。そこで、3.1、3.2では操作後のPDGは何が変化するのかを整理し、プログラマが変化されたくないものには何があるのかを整理した。対象物を操作するとき、プログラマから制約が与えられた場合に操作後のPDGにどのようなものがあるかを整理する必要がある。そこで、制約が全くない場合の操作後のPDGの候補を整理する。なぜならば制約のない候補は制約を与えた場合の候補を包含していると考えられるからである。

表3 操作に対する候補

対象物＼タイプ		ノード			データ依存関係リンク
		文中に変数の定義と参照がある文	文中で変数が参照されている文	while,if文などのプログラムの制御に関係のある文	
追加	複写	・Opのみ ・Op,if文 ・Op,代入文	・Opのみ ・Op,代入文 ・Op,代入文,if文	・Opのみ ・Op,代入文 ・Op,代入文,if文	
追加	新規追加	・Opのみ ・Op,if文 ・Op,代入文	・Opのみ ・Op,代入文 ・Op,代入文,if文	・Opのみ ・Op,代入文 ・Op,代入文,if文	・Op1,再定義文削除 ・Op2,再定義文削除 ・Op1,再定義文の後にif文 ・Op2,再定義文の後にif文 ・代入文移動 ・変数参照文移動
削除		・Opのみ ・Op,代入文 ・Op,変数参照文	・Opのみ ・Op,変数参照文	・Op,子ノード移動 ・Op,子ノード移動,if文 ・Op,子ノード移動,if文,代入文	・Op1'のみ ・Op2'のみ,代入文 ・代入文 ・代入文移動 ・変数参照文移動

表3は制約がない場合のPDGの操作要求に対する操作後のPDGの候補を示す。表3でOpと書かれたものはその操作を行うことを意味する。表中のOp1は、参照先の変数ノードに参照変数を追加する操作であり、Op2は定義側の定義変数を参照側の参照変数に変える操作を意味する。Op1'は参照側ノードの変数を消去あるいは変更する操作を意味し、Op2'は定義側ノードの定義変数を変更する操作を意味している。

例えば、図1(b)PDGのPDGにおいて、"x=1"から"sum=sum+x"へのデータ依存関係リンクを削除することを考えてみよう。このリンクを削除するには、"sum=sum+x"の変数xを他の変数に書き換える、あるいは、"x=1"を"y=1"のように変数を他の変数に書き換えるなどがある。表3から制約のない場合には、操作後のPDGの候補は5つあるが、定義側ノード"x=1"から出力されるデータ依存関係リンクを保存したいという制約を与えることによって、参照側変数の変数の変更か、参照文の前に新たに代入文を追加するのか、あるいは、参照文を定義文よりも前に移動するのかといった候補に絞られる。

4.まとめ

本稿では、PDG上での変更操作が行える環境を作成するために、プログラマが行える操作タイプを整理し、その操作タイプによってPDGのどの部分が変化するのかを示した。その操作によってプログラマの変更されてほしくない部分が変化することがある。そこで、変更タイプに対して、制約を与えられる候補には何があるのかを整理した。最後に、変更タイプに対してどのような候補があるのかを分類した。これによって、PDG上でのプログラマの操作要求に対して操作後のPDGにはどのようなものがあるかが提示できる環境を作成できる。

謝辞

本研究の一部は、稲盛財団の支援による。

参考文献

[1] K.Ottenstein and L.Ottenstein:The program dependence graph in a software development environments,ACM SIGPLAN Notices,Vol.19,No.5,pp.177-184(May 1984)
[2] S.Horwitz,T.Reps and D.Binkley:Interprocedual Slicing Using Dependence Graphs,ACM Trans. Programming Languages and Systems,Vol.12,No.1,pp.35-46(Jan. 1990)